```
      5
1 0
describe
```

apl utility programs

this workspace provides a collection of apl idioms and pseudo-primitives which are useful in a variety of application contexts.

apl idioms are single expressions which perform a single operation. they are most often used as a line, or part of a line, of an apl function. the function <idioms> in this workspace will list a collection of apl idioms which are separated into 8 categories.

apl pseudo-primitives are defined apl functions. their input comes from arguments, and their output is given through results returned. they are apl functions which can stand alone. they are usually either variants of existing apl primitives, or functions which work as complements to apl primitives.
the variable <summary> will list all of the pseudo-primitives which are collected in this workspace

```
idiomfns
**package**
idioms1

apl idioms -- structural

((vector⍳vector)=⍳⍴vector)/vector                ⍝ select unique elements

(1,1↓vector≠¯1⌽vector)/vector                    ⍝ select unique elements from
                                                   a sorted vector

(1 1⍉<\matrix∧.=⍉matrix)⌿matrix                  ⍝ select unique rows

(1,1↓∨/matrix≠¯1⊖matrix)⌿matrix                  ⍝ select unique rows from
                                                   a sorted matrix

(1⌈¯2↑⍴array)⍴array                              ⍝ make a row-oriented matrix

(1⌈2↑⍴array)⍴array                               ⍝ make a column-oriented
                                                   matrix

((⍴vector),1)⍴vector                             ⍝ make a 1 column matrix from
                                                   a vector

(1,⍴vector)⍴vector                               ⍝ make a 1 row matrix from
                                                   a vector

(((0 1×⍴mat2)⌈⍴mat1)↑mat1),[1]
      ((0 1×⍴mat1)⌈⍴mat2)↑mat2                   ⍝ catenate 2 matrices along
                                                   dimension 1

(((1 0×⍴mat2)⌈⍴mat1)↑mat1),[2]
      ((1 0×⍴mat1)⌈⍴mat2)↑mat2                   ⍝ catenate 2 matrices along
                                                   dimension 2

,vec1,[⎕io+.5] vec2                              ⍝ interleave 2 vectors of same
                                                   length
```

¯1⍤<,' ',vector ⍝enclose the vector individually
vector←'a b c'

```
     5

(lo-1)+ι1+hi-lo                              ⍝ integers between lo and hi
                                               limits

(vec1,vec2)[⍋⍋bool]                          ⍝ mesh 2 vectors according to
                                               boolean vector

x,[⎕io-(≁\2ρ∧/2>(ρρx),ρρy)/.5 1] y           ⍝ catenate/laminate 2 arrays
                                               to form rows

x,[⎕io+(≁\2ρ∧/2>(ρρx),ρρy)/.5 1] y           ⍝ catenate/laminate 2 arrays
                                               to form columns

(,b,[1.5] 1)/,b,[1.5] ~b                     ⍝ generate expansion vector
                                               from boolean vector
```

```
idioms2

apl idioms -- sorting and searching

vector[⍋vector]                              ⍝ sort a vector

matrix[⍋(1+ρkey)⊥⍉keyιmatrix;]               ⍝ sort the rows of a matrix

⎕io++⌿∧⍀~matrix1∧.=⍉matrix2                  ⍝ first index of rows in a
                                               matrix

(vector=scalar)/ιρvector                     ⍝ all indices of a scalar in
                                               a vector

⌊/vectorιelements                            ⍝ first occurrence of several
                                               elements in a vector

(scalar∊vector)×vectorιscalar              ⍝ first occurrence of or zero

(vector≠scalar)ι1                            ⍝ first index in vector which
                                               is not scalar

vector[⍋keyιvector]                          ⍝ sort vector in order
                                               specified by key
```

idioms3

apl idioms -- formatting

```
(+/∧\' '=array)⌽array                        ⍝ left justify
(1-(array=' ')⊥1)⌽array                      ⍝ right justify
width↑((⌊-/.5×width,⍴text)⍴' '),text         ⍝ center
(∨\' '≠text)/text                            ⍝ remove leading blanks
(⌽∨\' '≠⌽text)/text                          ⍝ remove trailing blanks
((text≠' ')∨text≠1⌽text)/text                ⍝ remove duplicate, adjacent
```

```
    5
                                                              blanks

((2×⍴text)⍴1 0)\text                                        ⍝ double space text
```

idioms4
apl idioms -- predicates

```
0=⍴vector                                                   ⍝ empty vector?
```

5

```
0∊⍴array                                     ⍝ empty array?

∧/,array∊0 1                                 ⍝ boolean array?

∧/,array=⌈array                              ⍝ integer array?

0=1↑0⍴array                                  ⍝ numeric array?

∧/elements∊set                               ⍝ all members of a set?

∨/elements∊set                               ⍝ any members of a set?

∧/vector1=vector2                            ⍝ 2 vectors identical?

vector1[⍋vector1]∧.=vector2[⍋vector2]        ⍝ 2 vectors permutations
                                               of each other?

∧/[axis] array=⌈\[axis] array                ⍝ array in ascending order
                                               along certain dimension?

∧/[axis] array=⌊\[axis] array                ⍝ array in descending order
                                               along certain dimension?
```

```
idioms5
apl idioms -- boolean shifts

            x← 0 1 0 1 1 0 0 1 0 1 1 1 0 0 0 1

x∧¯1↓0,x       0 0 0 0 1 0 0 0 0 0 1 1 0 0 0 0 ⍝ one and previous is one

x∧~¯1↓1,x      0 1 0 1 0 0 0 1 0 1 0 0 0 0 0 1 ⍝ one and previous is zero

x∧1↓x,0        0 0 0 1 0 0 0 0 0 1 1 0 0 0 0 0 ⍝ one and next is one

x∧~1↓x,1       0 1 0 0 1 0 0 1 0 0 0 1 0 0 0 0 ⍝ one and next is zero

(~x)∧¯1↓0,x    0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 0 ⍝ zero and previous is one

~x∨¯1↓1,x      0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 ⍝ zero and previous is zero

(~x)∧1↓x,0     1 0 1 0 0 0 1 0 1 0 0 0 0 0 1 0 ⍝ zero and next is one

~x∨1↓x,1       0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 ⍝ zero and next is zero


x≠¯1↓(1↑x),x   0 1 1 1 0 1 0 1 1 1 0 0 1 0 0 1 ⍝ previous bit is different

x=¯1↓(~1↑x),x  0 0 0 0 1 0 1 0 0 0 1 1 0 1 1 0 ⍝ previous bit is same

x≠1↓x,¯1↑x     1 1 1 0 1 0 1 1 1 0 0 1 0 0 1 0 ⍝ next bit is different

x=1↓x,~¯1↑x    0 0 0 1 0 1 0 0 0 1 1 0 1 1 0 0 ⍝ next bit is same
```

idioms6
apl idioms -- boolean scans

```
         x←1 1 1 0 1 1 0 1 0 0 0 1
∧\x        1 1 1 0 0 0 0 0 0 0 0 0 ⍝ ones until first zero from left

         x←1 1 1 0 1 1 0 1 0 0 0 1
~∧\x       0 0 0 1 1 1 1 1 1 1 1 1 ⍝ ones after first zero from left

         y←0 0 0 1 0 0 1 0 1 0 0 1
∨\y        0 0 0 1 1 1 1 1 1 1 1 1 ⍝ zeros until first one from left

         y←0 0 0 1 0 0 1 0 1 0 0 1
~∨\y       1 1 1 0 0 0 0 0 0 0 0 0 ⍝ zeros after first one from left

         z←0 0 1 1 1 0 0 1 0 0 1 0
<\z        0 0 1 0 0 0 0 0 0 0 0 0 ⍝ first one from left

         x←1 1 1 0 0 1 1 0 0 1 1 1
⌽∧\⌽x      0 0 0 0 0 0 0 0 0 1 1 1 ⍝ ones until first zero from right

         x←1 0 0 1 1 0 0 1 1 1 1 1
⌽~∧\⌽x     1 1 1 1 1 1 1 0 0 0 0 0 ⍝ ones after first zero from right

         y←1 1 1 0 0 1 0 1 0 0 0 0
⌽∨\⌽y      1 1 1 1 1 1 1 1 0 0 0 0 ⍝ zeros until first one from right

         y←1 1 1 0 0 1 0 1 0 0 0 0
⌽~∨\⌽y     0 0 0 0 0 0 0 0 1 1 1 1 ⍝ zeros after first one from right

         z←0 0 1 0 0 1 1 1 0 0 0 0
⌽<\⌽z      0 0 0 0 0 0 0 1 0 0 0 0 ⍝ first one from right

         v←0 1 0 0 0 1 0 0 1 0 1 0
≠\v        0 1 1 1 1 0 0 0 1 1 0 0 ⍝ boolean finite state machine

         v←0 1 0 0 0 1 0 0 1 0 1 0
v∨≠\v      0 1 1 1 1 1 0 0 1 1 1 0 ⍝ ones and ones between ones
```

```
      5
        v←0 1 0 0 0 1 0 0 1 0 1 0
 (~v)∧≠\v 0 0 1 1 1 0 0 0 0 1 0 0 ⍝ ones between ones
```

idioms7

apl idioms -- numerical

```
matrix×(⍴matrix)⍴vector                        ⍝ multiply vector times
                                                 rows of matrix

matrix×⍉(⌽⍴matrix)⍴vector                      ⍝ multiply vector times
                                                 columns of matrix

(+/[axis] matrix)÷(⍴matrix)[axis]             ⍝ average along specified
                                                 axis

array-0,[axis] (-axis=⍳⍴⍴array)↓array          ⍝ succesive first differences
                                                 along specified axis

1|array                                        ⍝ fractional part of number

0 1⊤array                                      ⍝ split integer and fractional
                                                 part of numbers

1+⌊base⍟number                                 ⍝ number of digits needed to
                                                 represent number in base

(10*n)|number                                  ⍝ keep n digits from right

⌊number÷10*n                                   ⍝ delete n digits from right

⌊.5+numbers                                    ⍝ round to nearest integer

(10*-n)×⌊.5+numbers×10*n                       ⍝ round to n digits right
                                                 of decimal point
```

idioms8

apl idioms -- object generators

```
(n,n)ρ1,nρ0                                     ⍝ identity matrix
(ιn)∘.≥ιn                                       ⍝ upper triangular matrix
(ιn)∘.<ιn                                       ⍝ lower triangular matrix
integerΔvector∘.≥ι⌈/integerΔvector              ⍝ prefix 1's
integerΔvector∘.≥⌽ι⌈/integerΔvector             ⍝ suffix 1's
(2*n)⊤0,ι¯1+2*2*n                               ⍝ truth table for n variables
```

summary

apl pseudo-primitives - variants and adjuncts to apl primitive functions

PRIMITIVE	FUNCTION	FEATURE
+	Δplus	vector extension to rows and columns
−	Δminus	vector extension to rows and columns
×	Δtimes	vector extension to rows and columns
÷	Δdiv	vector extension to rows and columns, commercial division
⌈	Δmax	vector extension to rows and columns
⌊	Δmin	vector extension to rows and columns
∧	Δand	vector extension to rows and columns

5

∨	∆or	vector extension to rows and columns
?!	∆perm	all permutations of ιn
?!	∆comb	all combinations of m elements of ιn
[;]	∆move	permutation with indexed fetch
[;]	∆from	matrix indexing
[;]←	∆erplc	element replace by value
[;]←	∆srplc	string replace by value
⍴ d	∆mfv	matrix from vector, structure determined by value
⍴ d	∆vfm	vector from matrix, structure determined by value
⍴ d	∆reshape	generalized reshape by value
, m	∆mat	make array into a matrix
, d	∆join	extend catenate/laminate along [1]
, d	∆link	extend catenate/laminate along [2]
ι m	∆thru	integers between left and right argument
ι m	∆between	integers between pairs of integers
ι m	∆integers	index generator for vectors
ι d	∆progindex	progressive index lookup
ι d	∆lastindex	last index of
ι d	∆find	string search- integer results
ι d	∆locate	string search- boolean results
ι d	∆indexof	matrix first index of
∊ d	∆memberof	matrix member of
/	∆repeat	replication with integer left argument
\	∆expansion	generates argument for expand
↑	∆lff	negative overtake with left first fill
↑	∆rlf	positive overtake with right last fill
↑↓	∆trim	take and drop based on values
⍋	∆gradeup	grade up matrix

apl pseudo-primitives - variants and adjuncts to apl primitive and derived functions

PRIMITIVE FUNCTION FEATURE

5

⌽ m̈	∆preverse	partitioned reverse
⌽ ā	∆protate	partitioned rotate
⍋	∆pgradeup	partitioned gradeup
⍒	∆pgradedown	partitioned gradedown
/	∆pmask	creates boolean vector argument for /
+/	∆pplusreduce	partitioned plus reduce
+\	∆pplusscan	partitioned plus scan
∧/	∆pandreduce	partitioned and reduce
∧\	∆pandscan	partitioned and scan
∨/	∆porreduce	partitioned or reduce
∨\	∆porscan	partitioned or scan
⌈/	∆pmaxreduce	partitioned max reduce
⌈\	∆pmaxscan	partitioned max scan
⌊/	∆pminreduce	partitioned min reduce
⌊\	∆pminscan	partitioned min scan
=\	∆peqscan	partitioned equal scan (boolean)
≠\	∆pneqscan	partitioned not equal scan (boolean)
<\	∆pltscan	partitioned less than scan (boolean)
≤\	∆pleqscan	partitioned les than or equal scan
⍎	∆nums	convert characters to numbers
⍕	∆format	format numbers, fill overflows with *
⍕	∆fmtblank	format numbers, blank out zeros
⍕	∆fmtcommas	format numbers, insert commas
⍕	∆fmtleft	format numbers, left justify
⍕	∆fmtminus	format numbers, use middle minus
⍕	∆fmtpic	format numbers using picture
⍕	∆fmtsmall	format numbers into smallest space
⍕	∆fmtzero	format numbers, zero fill